PB-216 714

A PILOT PROBLEM-ORIENTED COMPUTER LANGUAGE FOR HYDROLOGY AND HYDRAULIC ENGINEERING ('HYDRO')

G. Bugliarello, et al

Carnegie Institute of Technology
Pittsburgh, Pennsylvania

15 August 1966

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

WP-00415

PB 216 714

A PILOT PROBLEM-ORIENTED COMPUTER LANGUAGE FOR HYDROLOGY AND HYDRAULIC ENGINEERING ("HYDRO").

G. Bugliarello, W. D. McNally, J. T. Gormley, and J. T. Onstott

Carnegie Institute of Technology, Pittsburgh, Pa.

Paper presented at the 47th AGU Meeting, Washington, D.C., April 1966

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

August 15, 1966

Hl

ABSTRACT

The philosophy, specifications and operational characteristics of a problem-oriented computer language, HYDRO, for performing hydrological and hydraulic engineering analyses are presented. The present version of HYDRO is intended as a pilot for further development of the concept of problem-oriented languages in the field of water resources. It encompasses the areas of precipitation analysis, hydrograph analysis, open channel hydraulics, flood routing, and frequency analysis.

From the user's viewpoint, the most significant characteristics of the language are the extreme ease with which a person totally unskilled in computer programming can learn in a very short time to perform computer analyses in the areas covered by the language, and the possibility of stringing long series of commands. The ultimate potential of the language is discussed, together with the alternatives and issues arising in a comprehensive development aimed at defining the most advantageous man-computer interface in the area of water resources analyses.

INTERFACE BETWEEN THE COMPUTER AND THE WATER RESOURCES FIELD

The increasingly pressing needs for effective means of analyzing and controlling water resources problems make it imperative to use to the fullest advantage the capabilities of computer machines. Determination of the optimal man-computer interface in water resources is a many-faceted and today largely unexplored problem involving a variety of goals and possible approaches.

This paper describes one possible aspect of the interface, the design of a problem-oriented computer language, that is, a language with commands

and data format specifically geared to the field of water resources.

A content-oriented language represents, in a sense, the latest step in the evolution of communication between the user and the digital computer - an evolution that, as shown in Figure 1, has progressed through binary, octal, machine and assembly languages to the present alphanumeric, "universal" languages such as ALGOL or FORTRAN. This evolution has shifted increasingly the burden of programming from the user to the language. In a problem-oriented language, each instruction conveys a larger degree of information - and hence is more synthetic - than an instruction in a universal language, since it is specific to a particular field. Furthermore, the problem-oriented language takes care of all - or most - of the housekeeping of the program and organizes the translation from the language used by the user to the subsequent stages in the chain of evolution shown in Figure 1.

Fig. 1

An immediately obvious advantage is that a user can learn to program proficiently in his particular field in a very short time - of the order of hours - without any need to know an alphanumeric language such as FORTRAN. As will emerge later, this is by no means the only or the most important advantage of a problem-oriented language.

Several problem-oriented languages are already in existence for different fields of engineering e.g., the COGO language for surveying, the STRESS language for the analysis of frame structures, and a host of languages for the simulation of problems in dynamics and automotive design.

The problem-oriented language for water resources presented in this paper, HYDRO, stems from an idea first proposed in 1961 (1) when a preliminary survey was made of the computer procedures deemed necessary for a basic and continuous coverage of the field of hydraulic engineering.

The concept of the language, and its implications were amplified in subsequent discussions (2)(3).

As it stands today, the language is the result of a four-year pilot effort, intended primarily to explore the issues arising in the development of a comprehensive problem-oriented language for the field of water resources, and to provide a framework for rapid future expansion.

GENERAL DESIGN PHILOSOPHY

The development of a problem-oriented language provides a great deal of latitude for the designer. Several approaches can be followed; it is difficult, a priori, without having proceeded to a considerable extent in the design of the language, to decide which is the most favorable one. The latitude of apporach is particularly ample in the area of hydraulic engineering and water resources, where most of the problems are initially ill defined, in the sense that they are susceptible to more than one solution.

Furthermore, the design of a problem-oriented language cannot be crystallized either in time or in scope. Changes both in hardware and in software, as well as the development of new techniques for the analysis of water resources problems tend to make the optimum interface between the user and the computer a dynamic rather than a static one.

A first choice facing the designer is whether or not to subdivide the water resources field into several subfields, such as frequency analysis, precipitation analysis and channel flow, and to provide for each a problem-oriented language, with a structure and data format most suitable to it.

Subdivision has merit when intercommunication between the different

subareas is limited. It has been followed, e.g., in the M.I.T. Integrated Civil Engineering System (ICES), which is an assembly of different sublanguages for surveying (COGO), structural frame analyses (STRESS), soil stability analyses (SEPOL), etc. (4). The ICES components are integrated into a system by a general compiler which understands the content of the component subcompilers and makes them operate jointly.

In the case of water resources there is a considerable amount of exchange between the possible subareas entering into the solution of a typical problem. For instance, the determination of design spillway flows may begin, characteristically, with the analysis of precipitation, and proceed through flood routing (which in turn requires open channel flow) and frequency analysis. If separate compilers were used for each of these subareas, the user would be required to learn the programming instructions for each in order to perform a simple, routine analysis. Another difficulty inherent in this approach would be how to draw the interfaces between different subareas which, as accomplished by flood routing and open channel flow, are overlapping rather than sharply defined. Furthermore, if subfields are defined on the basis of type of problem and corresponding data structure, their number would be very large. In this respect, the situation in water resources is far more complex than, e.g., in structural frame analysis, where one deals with many different geometries, but where, once these have been described, the solution is given by the same equations for every frame structure.

All the above considerations militate against the subdivision of a water resources language into sublanguages. Thus, HYDRO has been designed as a comprehensive language, with a data structure valid for the entire range of water-resources problems it covers. This is a solution

in the direction of greater intergration and with potential for further expansion. The price one pays for such an approach is a slighly more general data structure, that is, a data structure less responsive to the needs of a given subarea. On the other hand, the user has to learn only one data structure and is given a tool of considerably more power and flexibility.

The flexibility in HYDRO is enhanced by the fact that the HYDRO processor - that is, the computer program converting HYDRO instructions into the universal language (ALGOL in this case) - has only the task of organizing the code that will carry out the solutions, but, unlike COGO or STRESS, does not contain solution procedures. These are contained in a library, and can thus be withdrawn, altered or added much more easily than if they were incorporated directly in the processor. Furthermore, the existence of a library of procedures separated from the processor,facilitates greatly the building of decision links into the language, and the use of the language in an on-line conversational mode.

The decision as to the basic structure of the language must be followed by several accessory ones: for example, the amount of logical decisions to be entrusted to the user versus that entrusted to the language, the size of the individual building blocks (procedures) of the language, and the criteria to be followed in maintaining, revising and augmenting the language. In the actual design of a procedure, it must also be decided how general or how specific the procedure should be: should the procedure be designed as a compromise acceptable to most potential users, or should there be instead several different procedures, each satisfying the needs of a particular group of users? In HYDRO, either approach has been followed, according to circumstances.

Each procedure in HYDRO has been generally designed according to the state of the art, that is, following established methods of hydraulic and hydrologic analysis. Several of these methods are of course an inheritance from the time when the only computational aids were the slide rule and sets of tables, and need to be replaced by more effective ones geared to the capabilities of a high speed, large storage computer; however, the development of new approaches is generally a very time-consuming process, which must be weighted against the advantages of speed in designing the language. Speed is important in order to build the language in a state of relative stability and cover rapidly the entire field. Once this is achieved, the language can be continuously improved and altered to embody new developments in the individual procedures.

The preceding considerations may well be alien to the normal background of the water resources specialist; it is nevertheless essential that he become conversant with them if he is to guide intelligently the development of an effective man-computer interface in his field.

DEVELOPMENT SPECIFICATIONS

The result of a lengthy series of decisions involving the general philosophy of HYDRO is embodied in a set of development specifications, which can be summarized as follows:

1. HYDRO should be much simpler to program than FORTRAN or ALGOL;
2. Commands and data should be oriented toward the field of water resources;
3. A broad area of water resources should be covered;

4. HYDRO should have flexibility to handle different types of problems;
5. HYDRO will have a library organized systematically for stringing of programs;
6. There should be automatic data transfer between commands;
7. In programming a problem involving a string of procedures or a single complex procedure, the user should be offered two alternatives: either to delegate to HYDRO the decision as to what procedure, or portion of a procedure, to proceed to (on the basis of the range of values assumed by given data variables) or to retain control of the decision-making process. Thus, in HYDRO'S pipe loop analysis program, currently under development, the user can input the information as to what pipes form a loop, or can leave such decisions to the program. Clearly, the more effective the guidance by the user, the greater is the time-saving in the execution of the program. On the other hand, control shifted to the user may lengthen the execution time since it increases the probability of errors in the input (e.g. an incorrect loop designation) or may be less precise, when the decision is based on the scanning and analysis of a large number of data;
8. Inputs and outputs should be both by card and teletype;
9. The language should be geared to the capabilities (i.e. memory capacities) of a broad range of computers;
10. The language should be easily augmentable by being designed in such a way as to facilitate the design of the procedures, the addition of new procedures to the library, or the removal and substitution of existing procedures;

11. In line with the two previous requirements, the language should have a modular compiler which is easy to expand, and be easily programmed in small compiler modules, so as to be suitable for different memory capacities.

SCOPE OF HYDRO

Figure 2 indicates the areas of the water cycle covered by the present version of HYDRO, as well as those where expansion is in progress. The coverage of the existing areas is not exhaustive, but provides a frame-
Fig. 2 work in the form of a basic set of operations, which, if necessary, can be easily expanded to provide a more comprehensive language.

There are at present five main areas covered by HYDRO: precipitation analysis, hydrograph analysis, open-channel hydraulics, flood routing, and
Table 1 frequency analysis. The procedures available in each are shown in Table 1, which lists actual HYDRO commands; the very descriptive nature of these commands is indicative of how HYDRO is geared to the language of the user.

The flow from program to program within each of the five areas in Table 1 is exemplified by the diagrams in Figures 3 to 7. Diagrams such as these constitute the "syntaxis" of HYDRO, as they indicate all the possible strings of instructions that can be formed, and the hierarchy between the different procedures (i.e., the degree of nesting characterizing each area).

Thus, Figure 4, showing the hierarchy in the area of hydrograph analysis, indicates that HYDRO provides five different approaches to the com-
Fig. 3-4 putation of a unitgraph, as a premise to the computation of a hydrograph. The first two involve the standard computation which reduces the ordinates of a runoff hydrograph for an isolated storm. These two differ in the com-

putation of the duration of rainfall excess - the first employing a constant infiltration rate, and the second varying the rate in a stepwise fashion. The third approach is computation of a triangular unitgraph, on the basis of strictly geometric data for the basin. As well known, this approach is used when insufficient data are available about the watershed. The fourth approach is a unitgraph bank method which involves averaging unitgraphs from many similar watersheds in order to form a synthetic unitgraph for a watershed for which data are not available. The fifth approach, usually followed when an isolated storm is not available, is a multiple regression unitgraph derived from a series of complex storms which have occurred relatively close to each other. The duration of the unitgraphs obtained with any one of these five approaches can be altered by the S-curve procedure. Once a unitgraph has been obtained, hydrographs can be computed - either standard hydrographs or, when the unitgraph was derived by regression techniques, regression hydrographs.

Fig.5 As a second example, Figure 5 is a flow diagram for the open-channel hydraulics segment of HYDRO. The upper portion of the diagram indicates the many different procedures which can be called to compute the hydraulic elements (area, wetted perimeter, topwidth, etc.) of channel sections.

The lower portion indicates higher order procedures which, in most cases, use data computed and retained by the various section elements procedures. For instance, after the computation of cross-section areas, mean discharges or velocities can be computed, depending upon the data available. Then, with areas and discharges as inputs, other procedures can be called to compute specific energy, momentum or pressure force at the channel sections, as well as critical or sequent depths. If the lengths

and slopes of channel reaches are known or computed, normal depths can be calculated, as well as backwater profiles on the basis of a single control point or of multiple control points. Similarly, other sequences of procedures can be called, leading to the computations of discharges, average channel roughness, normal or critical slopes, or friction head losses by means of the Manning equation. In all cases, the user has the option of entering the diagram at any point, either using data already available at hand, or computing needed data by calls on appropriate procedures of lower order.

Fig.6-7 It should be noted that the construction of hierarchical diagrams such as those in Figures 3 to 7 is per se a valuable byproduct of the design of the language, since it clarifies the logical structure of the field.

PROGRAMMING RULES

The rules for programming in HYDRO are extremely easy, and are explained in detail in a Users Manual (5). In essence, the simplest HYDRO program consists of:

1. a procedure call (e.g. NORMAL.DISCHARGE, DEPTH.AREA.DURATION.CURVES)
2. the description of the data, and
3. PROGRAM.END

In general, the outputs of one procedure can be retained as inputs to subsequent ones; thus, it is possible to write long strings as follows:

```
Procedure Call
Data
Procedure Call
Data
  ...
Program End
```

The data are placed on separate cards from the procedure calls and are

inputted in the format:

Variable Name ← Data;

Variable Name may represent both numeric and alphanumeric single variables and arrays. If it represents an array, several pieces of data are usually given following the left arrow, and separated by commas. Alphanumerical data are enclosed in asterisks.

EXAMPLES

Two programs will be presented in detail. The first is a program for determining the cross sectional area, the wetted perimeter, the width of the channel at the water surface (top widths), and the centroidal distance for the portion of a cross section bounded by the channel wall and the water surface.

These computations are to be carried out for several stations along the channel, and several water surface elevations at each station. The available data to this effect are the number of stations, defined by a beginning station and an end station, the maximum number of points needed to describe the polygonal shape of each cross section, the maximum number of areas for which the computations are to be carried out at any station, the number of points actually needed to describe each cross section, the number of water surface elevations for which computations are to be carried out at each station, the X and Y coordinates of the points describing each cross section, and the water surface elevations.

The flow chart for the computation is shown in Figure 8, in which the dotted arrows from block to block represent outputs retained as inputs to the subsequent block. The outputs obtained at each block are the cross-sectional

area, the wetted perimeter, the top width, and the centroidal distance. Figure 9 shows the cross sections for which the computations are to be carried out, and the water surface elevations at each station. The actual HYDRO program is shown in Figure 10, and is self-explanatory; the output is shown in Figure 11. It is worth noting that the ALGOL code generated as a result of the 26 lines of HYDRO code in Figure 10 is more than 400 lines in length - a telling indication of the synthesis made possible by HYDRO.

Figs. 8-11

The second example - considerably more complex in content than the first - involves the computation of a set of depth-area-duration curves for a multi-center storm over a basin. The computational procedure follows closely that suggested by the U. S. Weather Bureau (6). As indicated by the HYDRO flow chart in Figure 12, the solution is achieved through six procedures strung and run in series. The figure shows the order in which the procedures are called, and the inputs and outputs for each procedure:

1) Six-hour masscurves of rainfall are computed for each of the stations in the watershed from hourly rainfall masscurves.

2) The maximum rainfall at each station in subsequent six, twelve, eighteen, and twenty-four periods is computed.

3) A curve of rainfall depth versus surface area is computed from data planimetered from the isohyetal map. The curve relates the average depth of rain to areas surrounded by the various isohyetal lines on the map.

4) Depth-duration curves are computed for isohyets within individual zones of the isoheytal map, using the six-hour masscurves at individual stations, as well as planimetered areas containing such stations within given isoheyts.

5) Similar curves are computed for combinations of zones.

6) All depth-duration curves which have been computed are ordered according

to the size of the areas to which they pertain.

The map of the basin is shown in Figure 13, with a superimposed precipitation pattern for which the computations are carried out. Also given in the figure are the masscurves for rainfall at eight different raingage stations in the basin. The storm has two centers, and the isohyetal map has been separated accordingly in Figure 13 into two zones, A and B.

Figs. 12-15

The actual HYDRO program is given in Figure 14. The first procedure, SIX.HOUR.MASSCURVES, requires the majority of the data for the program. TOTAL11 gives the number of raingage stations in the watershed (8 stations) and TOTAL10 gives the number of hours of rainfall at each of these stations (20 hours). The station names are given by STATION1 and the rainfall zones in which each is located by ZONE1. Thus ZONE1 ← *A*, *A*, *A*, *A*, *B*, *B*, *B*, *B* ; means that the first four stations are in zone A and the remaining four stations are in Zone B.

The two-dimensional, mass input array variable gives the masscurves of rainfall at the eight stations, each masscurve containing 20 values. The data can be given sequentially without any need to denote by a special sign the end of the data for a given station.

The second procedure, MAXIMUM.STATION.PRECIPITATION, requires no additional inputs. The third, DEPTH.AREA.CURVE, requires planimetered areas of each isohyetal contour in each rainfall center of the composite isohyetal map.

The other procedures of the program are called in similar fashion, each followed by the data necessary for its execution. The detailed specifications of how the data for these procedures are to be inputted are contained in the Users Manual.

The final procedure, DEPTH.AREA.DURATION.CURVES, yields the results shown

in Figure 15. Once again, it is indicative of the synthesis achieved with HYDRO, that the number of code lines required for programming the same problem in ALGOL is approximately 800, versus approximately 50 in HYDRO.

In addition to the final outputs, the user receives the intermediate outputs indicated in the flow chart in Figure 12.

The first procedure, yields six-hour masscurves of rainfall at each raingage station; the second, the maximum 6, 12, 18, and 24 hour precipitation at each station; the third, the average depth of rainfall (AVDEPTH) over various size areas (AREA 2) in the watershed; the fourth and fifth, maximum depth duration curves (MAX3 and MAX4) for single rainfall zones and combinations of zones throughout the watershed.

THE HYDRO SYSTEM: CHARACTERISTICS AND OPERATION

The HYDRO system, i.e. the software assembly which accepts a HYDRO program,translates it into ALGOL code, and executes it, is represented schematically in Figure 16. Basically, the system is divided into two major parts: the working part, shown on the left of the diagram, and the compiler in storage, shown on the right. The working part consists in turn of three components: the compiler in use, the language library and an assembly area. All of these software components may be stored on either disc or tape.

Fig. 16

The compiler in use is called into operation whenever a program is submitted. Several sections of the compiler, though usually not all, are required in order to translate and compile the HYDRO code. The language library contains all the procedures of the HYDRO language, coded in ALGOL. When a user's HYDRO program is submitted, not all of the language library is called into the memory of the computer, but only those portions necessary

for the execution of the actual commands in the specific program. These portions are arranged in the proper order for execution in the assembly area, which is merely a general working space.

The compiler in storage, also stored on disc or tape, is a duplicate of the compiler, used only when alterations are being made to the HYDRO system.

The present dimensions of the components and subcomponents of the HYDRO system are given in Figure 17. Adding to the dimensions of the HYDRO software components the memory required for normal housekeeping in the computer, and for the execution of the ALGOL language, a total of approximately 32,000 words of core memory capacity are needed for the implementation of the present HYDRO system. However, it must be emphasized that, if necessary, the system can be broken into smaller segments and made thus suitable for smaller machines. As noted in Figure 17, the largest assembled program run to date has consisted of 1,000 ALGOL card images corresponding to 12,000 words of internal storage to run. This is very close to the maximum sized program which can be run without segmentation on the 32,000 word Bendix G-20 machine on which HYDRO now operates at Carnegie.

Fig. 17

The working of the HYDRO system is shown in very simplified fashion in Figure 18, where the order is indicated in which the assembly of a HYDRO program is carried out. The transferral from the compiler in use to the computer core, indicated by arrow 1, takes place section by section throughout the seven sections of the compiler. Frequently, depending on the type of problem to be solved, i.e. on the procedures being called, not all sections need to be transferred from the compiler to the computer core. Again, from the library only those sections are transferred to the computer core (arrow 3) that are needed for the execution of a given program.

Fig. 18

While the segments of the compiler loaded into the core are being run, the ALGOL compiler must also be in the core and operate, since the HYDRO compiler segments are coded in ALGOL. After the programs in the HYDRO library, also coded in ALGOL, have been arranged in proper order on the assembly area, the HYDRO compiler can be removed from the core, and the assembled program loaded into it, to run. Again, the ALGOL compiler must be retained in core during the running of the program.

As can be expected, the computer time required for the execution of a HYDRO program is longer - usually from 20 to 40% - than the time required by a FORTRAN or ALGOL program for performing the same task. This modest increase in execution time must be weighted against the very considerable saving in programming time made possible by HYDRO, through its simplicity, elimination of debugging errors of the underlying alphanumeric code, and retention of data from commancd to command.

THE SIGNIFICANCE OF HYDRO

The significance of a language such as HYDRO is manifold. Unquestionably, one of the most important aspects is that for the first time a water resources specialist is given the capability to use the computer without any previous knowledge of programming languages such as ALGOL or FORTRAN, and to perform rather complex computer analyses without the assistance of a programming specialist. Thus, in a few hours, a user totally ignorant of languages such as ALGOL or FORTRAN - as many water resources specialists still are today - can learn to program effectively in HYDRO.

Through the use of HYDRO, even a very small professional organization

can acquire the capability to conduct computer analyses which until now could be carried out only by much larger organizations with programming specialists on their staff. This capability can be further enhanced by the use of teletypes which, by means of time-sharing, can bring into the office of the user, for a very modest rental fee, the power of a large computer.

Larger, more sophisticated organizations, confronted with very simple tasks can also draw a signal advantage from HYDRO by using it as a simulation language, for the rapid performance of preliminary analyses and design studies. These, of course, can be supplemented by more specific ad hoc programs, written in a language such as ALGOL or FORTRAN. If these ad hoc programs are likely to be used repeatedly, they may profitably be incorporated by the user in his copy of HYDRO (either as additional procedures, or as substitutes for existing procedures), by taking advantage of HYDRO's open strucrure. In this case, if the developer of the new program is agreeable to it, the program can become available to, and immediately usable by, all other users of HYDRO - a factor that would contribute to the enrichment and expansion of the HYDRO system and to the reduction of the large amount of duplication in programming effort characterizing today the water resources field. The amount of duplication can be further reduced if henceforth any new program in this field, whether or not it is intended for incorporation in HYDRO, is written in a way compatible with the requirements of HYDRO. The essential individuality of the new programs would not be invalidated, but their unified format and data structure would make them easily interchangeable and accessible to any user in the field. This is a significant potential benefit from a language such as HYDRO, which would accrue even if the language were never to become widely operational.

Yet another significant implication of HYDRO is its potential as an

instructional tool. HYDRO makes it practical, for the first time, to assign to the student complex water resources problems, and to have him work on the essence of the problem, without being sidetracked by programming intricacies, delayed by debugging, and discouraged by repeated passes through the computer He can thus retain the perspective of the problem, and test rapidly the consequences of changes in data or method of attack. Similar advantages can be reaped in a demonstration situation - such as in a classroom - if HYDRO is programmed on a time-sharing teletype.

Finally, no matter how conservative a view one may take of HYDRO, it must be conceded that its greatest significance and justification lies in the fact that it is the first attempt to develop an effective and simple man-machine interface in the field of water resources, and in the persepctive thereby gained for further developments. To paraphrase Allen Newell, in computer science - one cannot argue about the merits of one approach without having developed it first.

STRENGTHENING HYDRO

By way of conclusions, it is worthwhile to discuss briefly some of the means whereby the effectiveness of HYDRO as it stands today can be furthered.

In the first place, as suggested previously, the scope of HYDRO should be amplified in order to include other areas in the field of water resources and to strengthen the coverage of those areas currently incorporated in the language. Such an expansion is already in process to a considerable extent.

Secondly, each of the existing procedures should be refined so as to embody the most advanced methodology, and thoroughly tested for the largest variety of situations (data) in each one of these areas.

To do so will require a continuous dialog between the developers of

HYDRO and the potential users. To this effect, it will also be necessary, of course, to provide versions of HYDRO based on languages other than ALGOL - such as FORTRAN II, FORTRAN IV and MAD. This is an easy and rather mechanical task.

The amplification and refinement of HYDRO are made possible by the flexible design of the processor which allows procedures to be easily removed from, or added to HYDRO. In the long run, it could be advantageous in this respect to devise a system which would operate on the HYDRO system and would enable a user, with a minimum of programming intervention, to add, delete or alter programs in the HYDRO language.

Another useful and easily achieved addition is to program and incorporate into HYDRO the Users Manual, so as to enable the language, if desired, to be programmed conversationally "on line."

If an expanded HYDRO system is to become of general utility in the solution of complex water-resources problems spanning a variety of content areas, combination with other problem-oriented languages for structural analyses, surveying, soil mechanics and for economic analyses may also be considered. Experience is today still too limited to assess the extent to which such a combination is desirable or effectively achievable. Clearly, however, the advantages of consolidation or compatibility with other languages are greater where there is considerable interaction between their respective content areas.

Finally, as the examples presented in this paper indicate, with HYDRO the only real burden left to the programmer is the unavoidable tedious task of inputting the long strings of data which generally characterize the water resources field. This burden can be alleviated through the use of the hardware capabilities for computer graphics, which are currently being introduced on

the market. The nature and structure of HYDRO are well suited to facilitate the match with such capabilities.

ACKNOWLEDGEMENTS

The research described in this paper has been conducted under support of Grant WP 00415, of the U. S. Public Health Service. The advice, throughout the course of the project, of Allen Newell, Institute Professor of Systems and Communication Sciences at Carnegie Institute of Technology, is gratefully acknowledged. Most of the material contained herein has been developed as part of a PhD thesis by W. D. McNally, and of M.S. theses by J. T. Gormley, W. D. McNally and J. T. Onstott. The programming assistance of D. Thomas is also gratefully acknowledged.

REFERENCES

1. Bugliarello, G.,"Toward a Synthetic Computer Language for Hydraulic Engineering," Proceedings, Xth General Meeting, International Association for Hydraulic Research, Belgrade, 1961.

2. Bugliarello, G.,"Electronic Computer Use in Scoping Power Projects," (Discussion), Journal of the Sanitary Engineering Division, ASCE, Vol. 87, July 1961.

3. Bugliarello, G.,"Man-Machine Communication in Civil Engineering," (Discussion), Journal Structural Division, ASCE, Vol. 90, ST1, p. 253-255, 1964.

4. Roos, D., ICES System Design, M.I.T. Press, Cambridge, Mass. (1966).

5. McNally, W. D., HYDRO Users Manual, Carnegie Institute of Technology, Pittsburgh, Pa. (1966)

6. U. S. Weather Bureau, Manual for Depth-area-Duration Analysis of Storm Precipitation . Technical Paper No. 1., Washington, D. C. (September 1946).

TABLE 1

PRESENT HYDRO PROCEDURES

PRECIPITATION ANALYSIS:

ARITHMETIC.RAINFALL.AVERAGE
THIESSEN.RAINFALL.AVERAGE
DOUBLE.MASS.BREAK.CHECK
RAINFALL.DOUBLE.MASS.ANALYSIS
MISSING.RAIN.STATION.AVERAGE
MISSING.RAIN.NORMAL.RATIO
NORMAL.ANNUAL.PRECIPITATION
SIX.HOUR.MASSCURVES
MAXIMUM.STATION.PRECIPITATION
DEPTH.AREA.CURVE
DEPTH.DURATION.SINGLE.ZONES
DEPTH.DURATION.COMBINED.ZONES
DEPTH.AREA.DURATION.CURVES

HYDROGRAPH ANALYSIS:

STD.UNITGRAPH.MAX.INFILT.RATE
STD.UNITGRAPH.VARY.INFILT.RATE
TRIANGULAR.UNITGRAPH
UNITGRAPH.BANK
SYNTHETIC.UNITGRAPH
MULTIPLE.REGRESSION.UNITGRAPH
S.CURVE.HYDROGRAPH
HYDROGRAPH
HYDROGRAPH.REGRESSION

OPEN CHANNEL HYDRAULICS:

CROSS.SECTION.AREA
WETTED.PERIMETER
TOPWIDTHS
CENTROIDAL.DISTANCE
HYDRAULIC.RADIUS
AVERAGE.MANNING.N
CONVEYANCES
ENERGY.COEFFICIENT
MOMENTUM.COEFFICIENT
SECTION.ELEMENTS - previous nine procedures with a single command
AREA.ALPHA - combination of cross-section area and energy coefficient
AREA.BETA - combination of cross-section area and momentum coefficient
AREA.CENDIST - combination of cross-section area and centroidal distance
AREA.HYDRAD.AVGMANN - combination of cross-section area, hydraulic radius and average Manning's n
DISCHARGE
MEAN.VELOCITY
REACH.LENGTH - lengths of reaches between stations
REACH.SLOPE - bed slopes of reaches between stations
SPECIFIC.ENERGY
MOMENTUM
PRESSURE.FORCE
NORMAL.DEPTH
CRITICAL.DEPTH
SEQUENT.DEPTH
NORMAL.DISCHARGE
CHANNEL.ROUGHNESS
SLOPES - normal or critical
FRICTION.HEAD.LOSS
FLOW.PROFILES.CONTROL.PT - flow profile between two control points
OPEN.CHANNEL.FLOW.PROFILES - flow profiles past many control points

TABLE 1
(continued)

FLOOD ROUTING:

FLOOD.ROUT.MUSKINGUM
FLOOD.ROUT.GENERAL - finite approximation and interpolation

FREQUENCY ANALYSIS:

FULL.SERIES
TIME.SERIES
PARTIAL.SERIES
LOW.ANNUAL.PARTIAL.SERIES
DATA.ORDERING
MAX.EVENT
MIN.EVENT
MEAN.EVENT
STANDARD.DEVIATION.OF.EVENTS
HISTOGRAM
DURATION.CURVE
HISTOGRAM.DURATION.CURVE
FREQUENCY.CURVE
HISTOGRAM.FREQUENCY.CURVE
GUMBEL.RECURRENCE.INTERVAL
STANDARD.RECURRENCE.INTERVAL

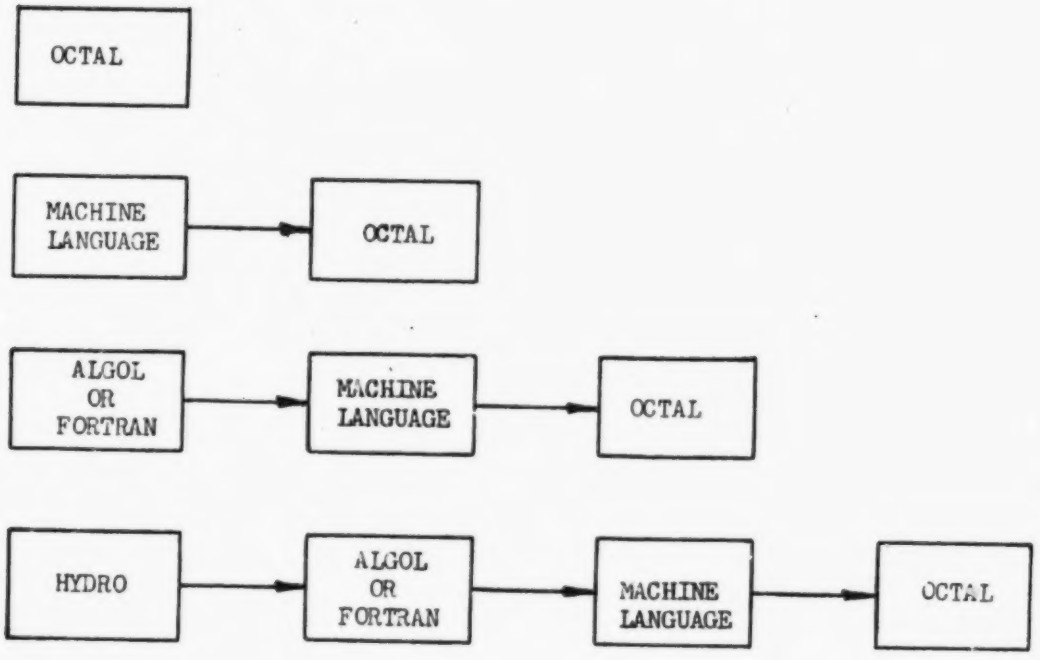


FIG. 1 EVOLUTION OF USER-ORIENTED COMPUTER LANGUAGES

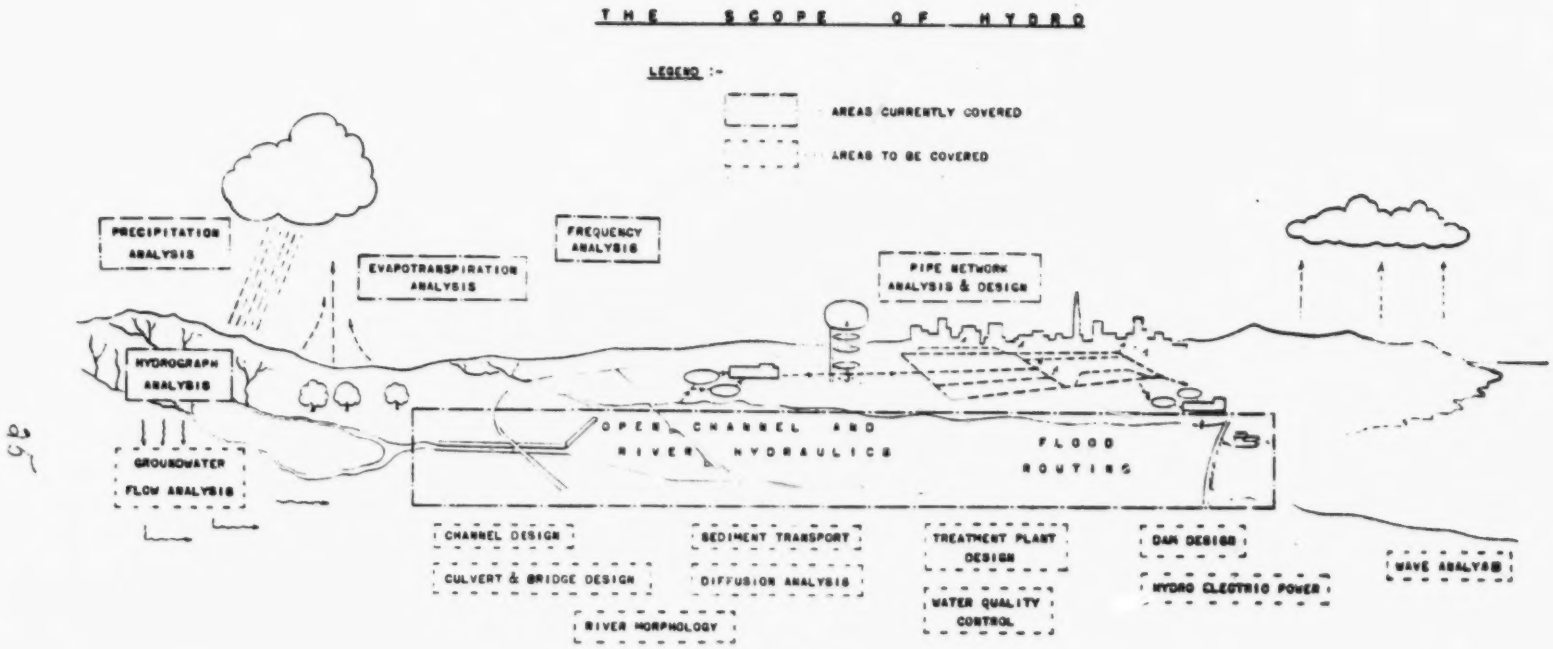


FIG. 2 THE SCOPE OF HYDRO

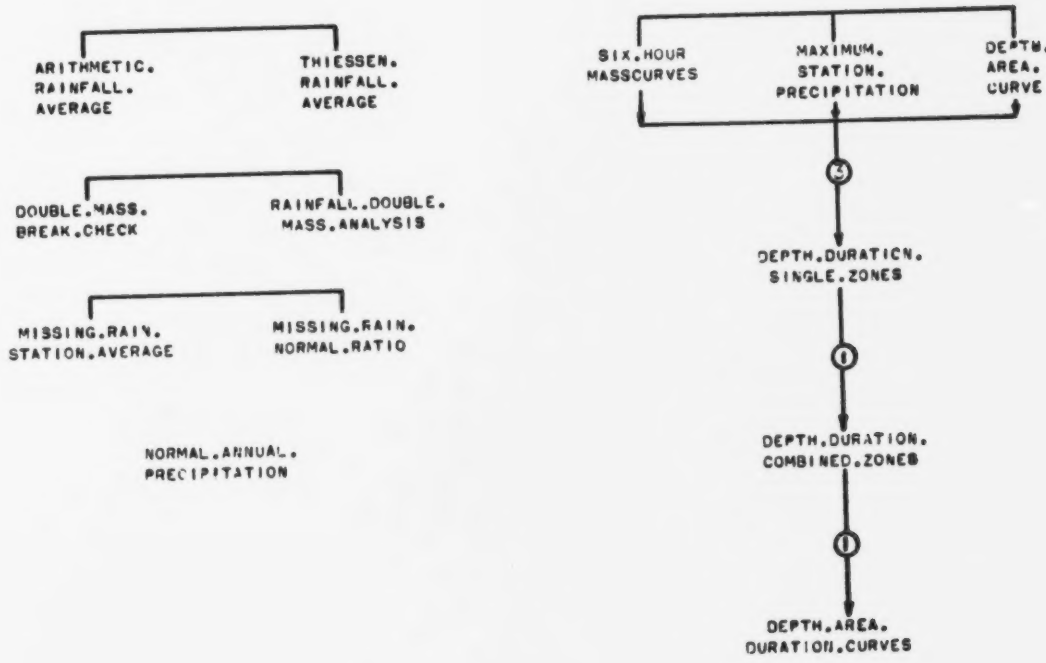


FIG. 3 PROGRAM HIERARCHIES: PRECIPITATION ANALYSIS

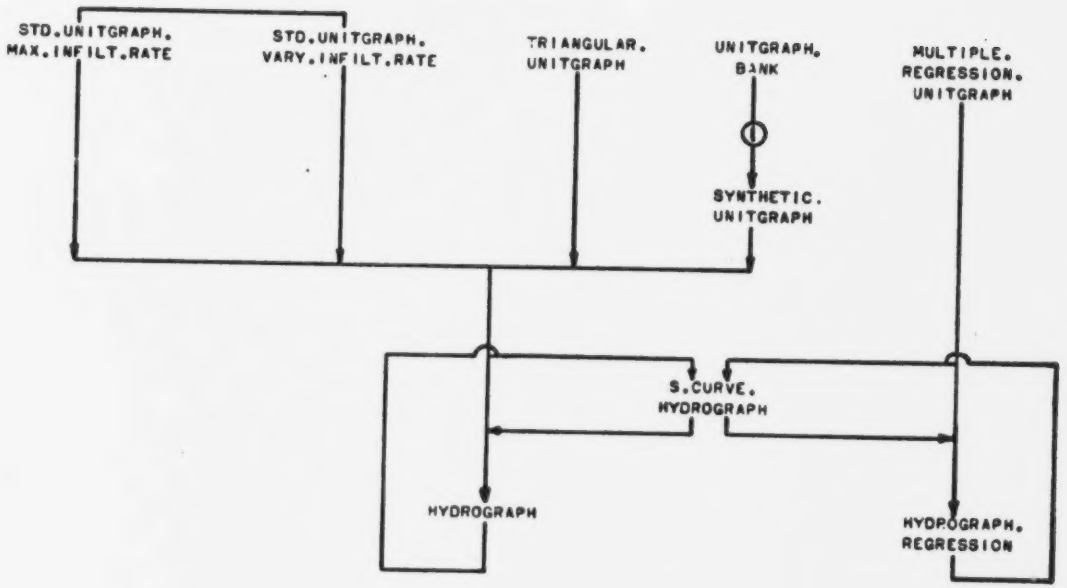
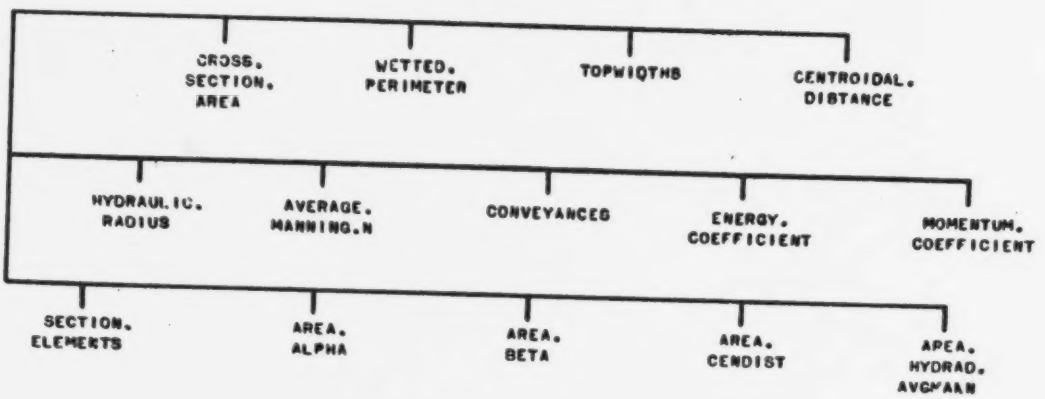


FIG. 4 PROGRAM HIERARCHIES: HYDROGRAPH ANALYSIS

A) SECTION ELEMENTS



B) CHANNEL HYDRAULICS

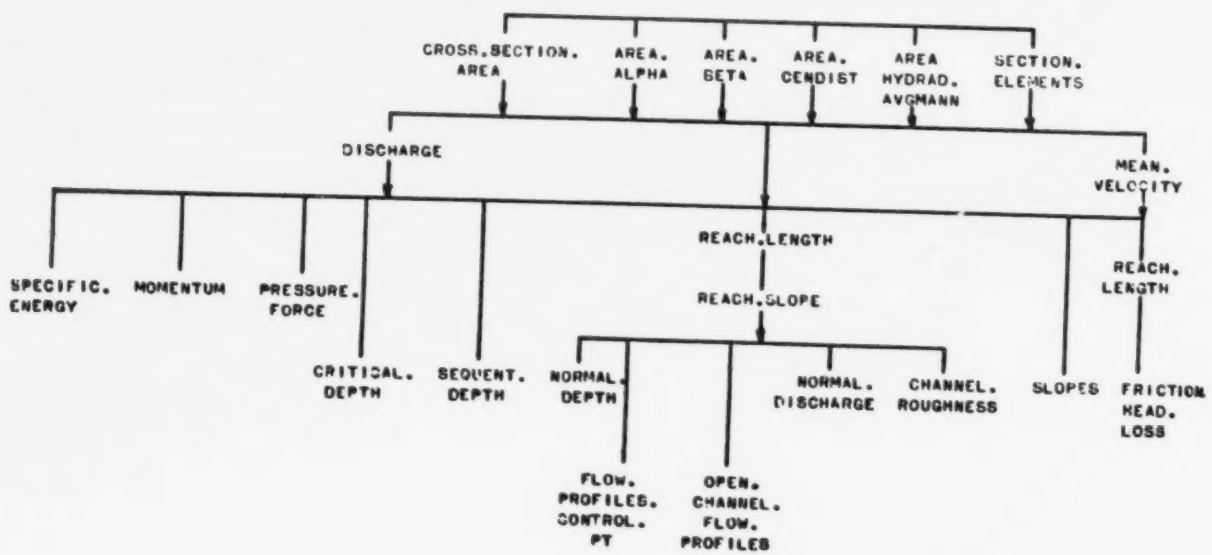


FIG. 5 PROGRAM HIERARCHIES: OPEN CHANNEL HYDRAULICS

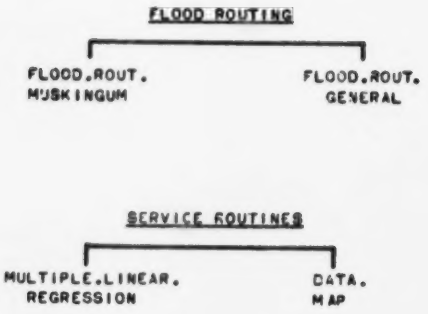


FIG. 6 PROGRAM HIERARCHIES: FLOOD ROUTING

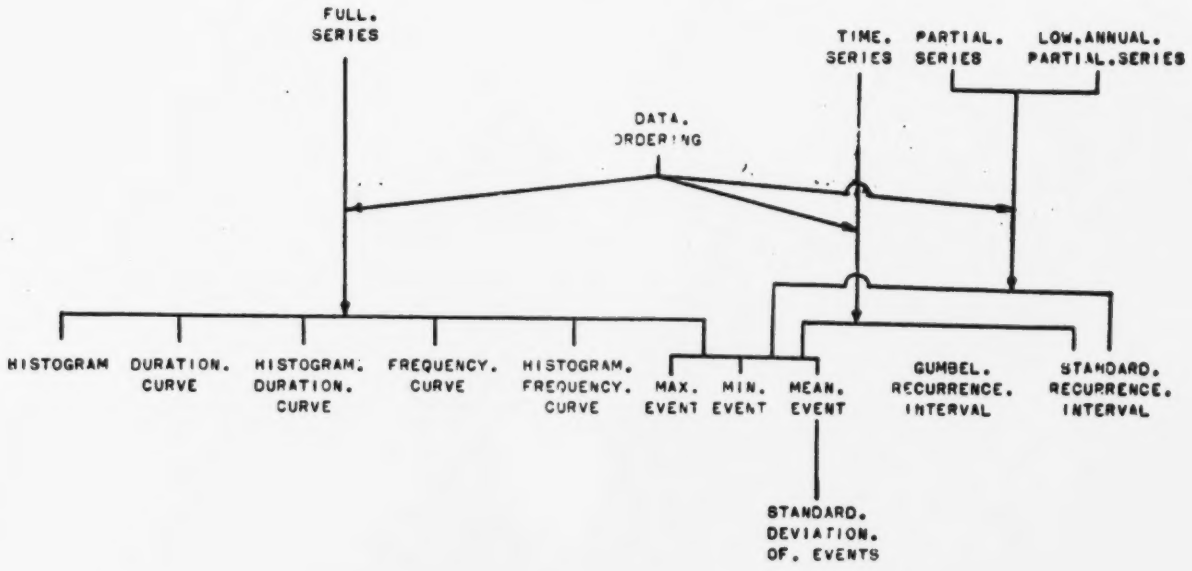


FIG.7 PROGRAM HIERARCHIES: FREQUENCY ANALYSIS

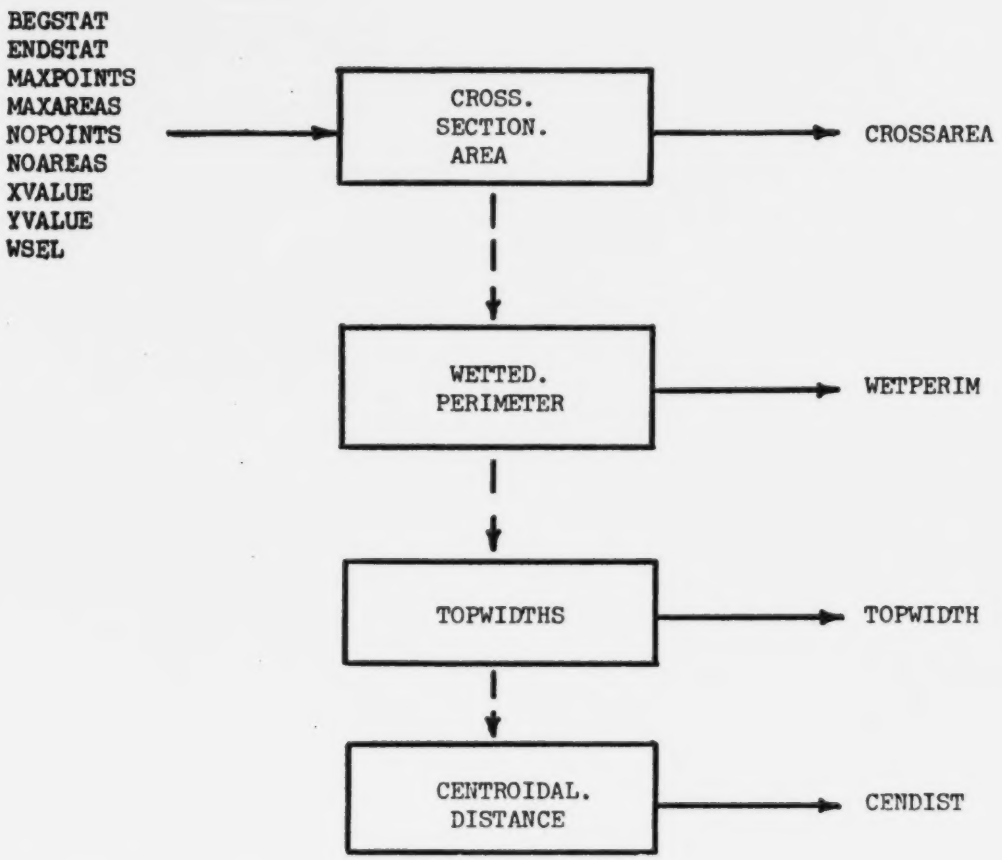


FIG. 8 EXAMPLE, OPEN CHANNEL HYDRAULICS: HYDRO FLOW CHART

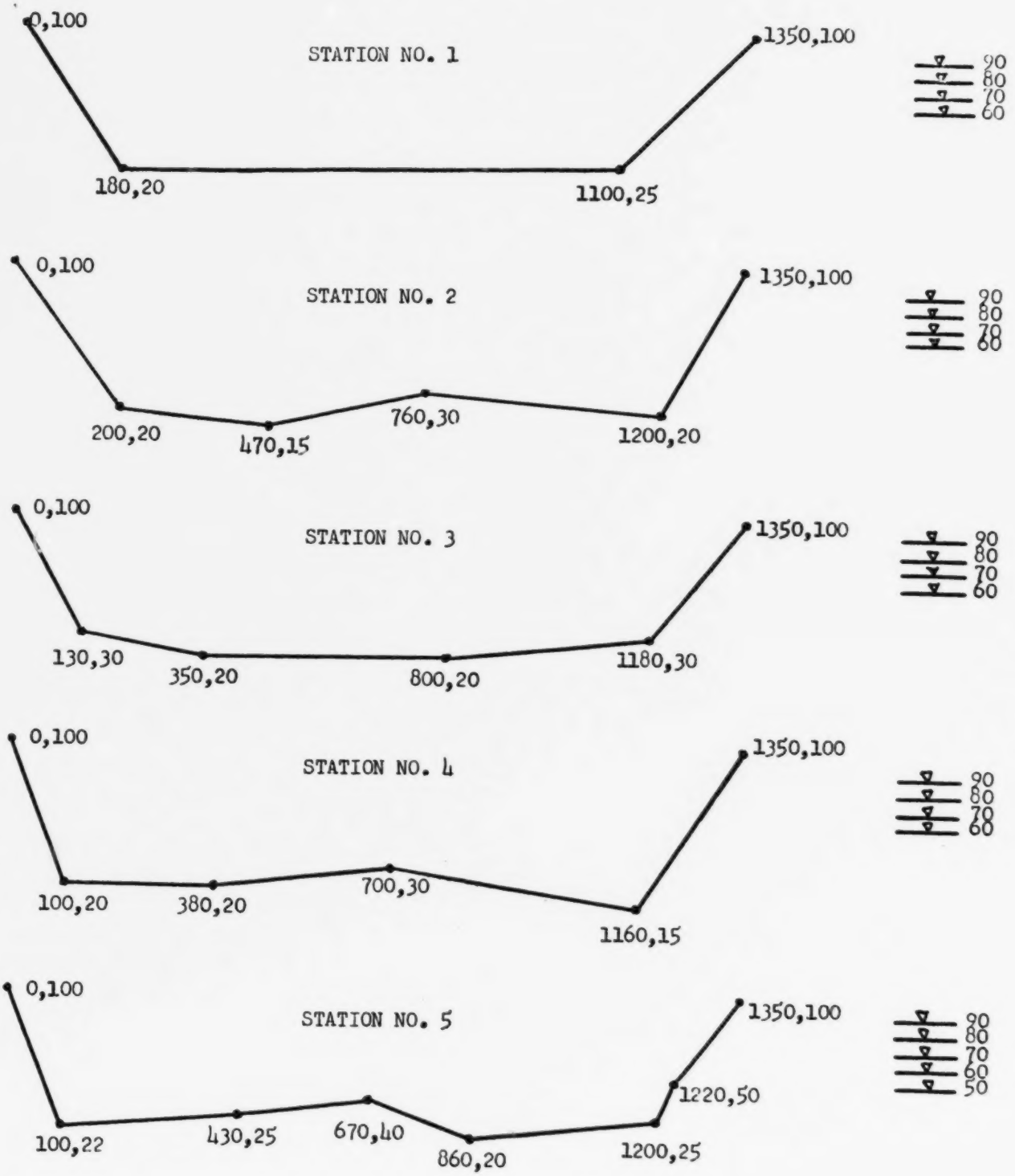


FIG. 9 EXAMPLE, OPEN CHANNEL HYDRAULICS: CROSS-SECTIONAL DATA

```
CROSS.SECTION.AREA
   BEGSTAT ← 1 ;
   ENDSTAT ← 5 ;
   MAXPOINTS ← 8 ;
   MAXAREAS ← 5 ;
   NOPOINTS ← 4, 6, 6, 6, 8 ;
   NOAREAS ← 4, 4, 4, 4, 5 ;
   XVALUE ← 0, 180, 1100, 1350, 0, 0, 0, 0,
            0, 200, 470, 760, 1200, 1350, 0, 0,
            0, 130, 350, 800, 1180, 1350, 0, 0,
            0, 100, 380, 700, 1160, 1350, 0, 0,
            0, 100, 430, 670, 860, 1200, 1220, 1350 ;
   YVALUE ← 100, 20, 25, 100, 0, 0, 0, 0,
            100, 20, 15, 30, 20, 100, 0, 0,
            100, 30, 20, 20, 30, 100, 0, 0,
            100, 20, 20, 30, 15, 100, 0, 0,
            100, 22, 25, 40, 20, 25, 50, 100 ;
   WSEL ← 60, 70, 80, 90, 0,
          60, 70, 80, 90, 0,
          60, 70, 80, 90, 0,
          60, 70, 80, 90, 0,
          50, 60, 70, 80, 90 ;
WETTED.PERIMETER
TOPWIDTHS
CENTROIDAL.DISTANCE
PROGRAM.END
```

FIG. 10 EXAMPLE, OPEN CHANNEL HYDRAULICS: HYDRO PROGRAM

STATION	WATER SURFACE ELEVATION	CROSS SECTION AREA	TOP WIDTH	WETTED PERIMETER	CENTROIDAL DISTANCE
1	60.00	38341.7	1126.7	1140.3	18.14
1	70.00	49887.5	1182.5	1199.7	22.78
1	80.00	61991.7	1238.3	1259.2	27.35
1	90.00	74654.2	1294.2	1318.6	31.85
2	60.00	41250.0	1175.0	1193.3	18.64
2	70.00	53218.7	1218.7	1241.4	23.31
2	80.00	65625.0	1262.5	1289.6	27.96
2	90.00	78468.7	1306.2	1337.8	32.56
3	60.00	40928.6	1178.6	1192.4	18.31
3	70.00	52928.6	1221.4	1239.8	23.02
3	80.00	65357.1	1264.3	1287.1	27.68
3	90.00	78214.3	1307.1	1334.5	32.31
4	60.00	42913.2	1210.6	1234.6	18.56
4	70.00	55193.4	1245.4	1275.1	23.31
4	80.00	67822.1	1280.3	1315.6	28.03
4	90.00	80799.3	1315.1	1356.1	32.73
5	50.00	26847.6	1155.9	1179.1	12.36
5	60.00	38600.6	1194.7	1223.2	17.07
5	70.00	50741.9	1233.5	1267.3	21.78
5	80.00	63271.4	1272.4	1311.5	26.47
5	90.00	76189.1	1311.2	1355.6	31.13

FIG. 11 EXAMPLE, OPEN CHANNEL HYDRAULICS: OUTPUT

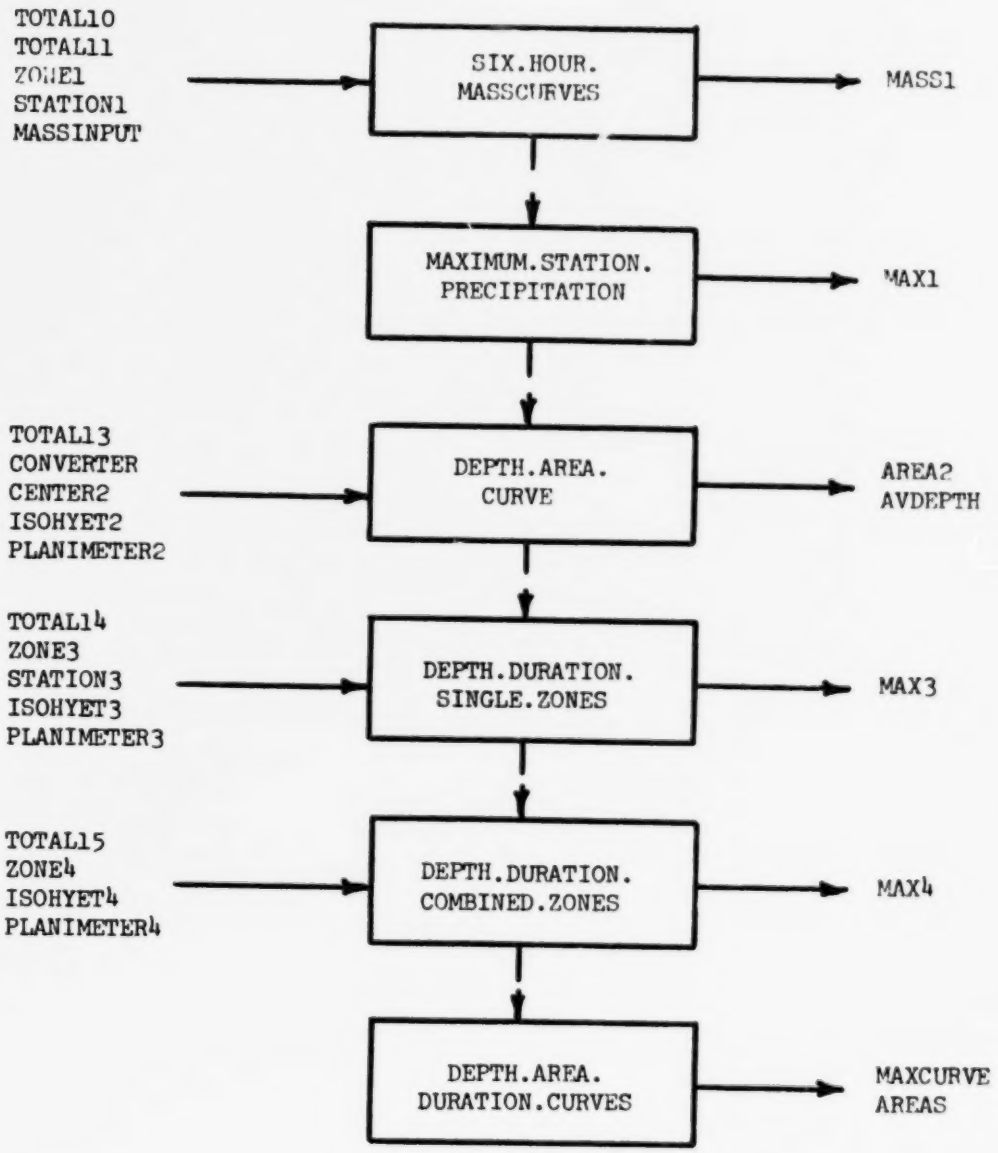
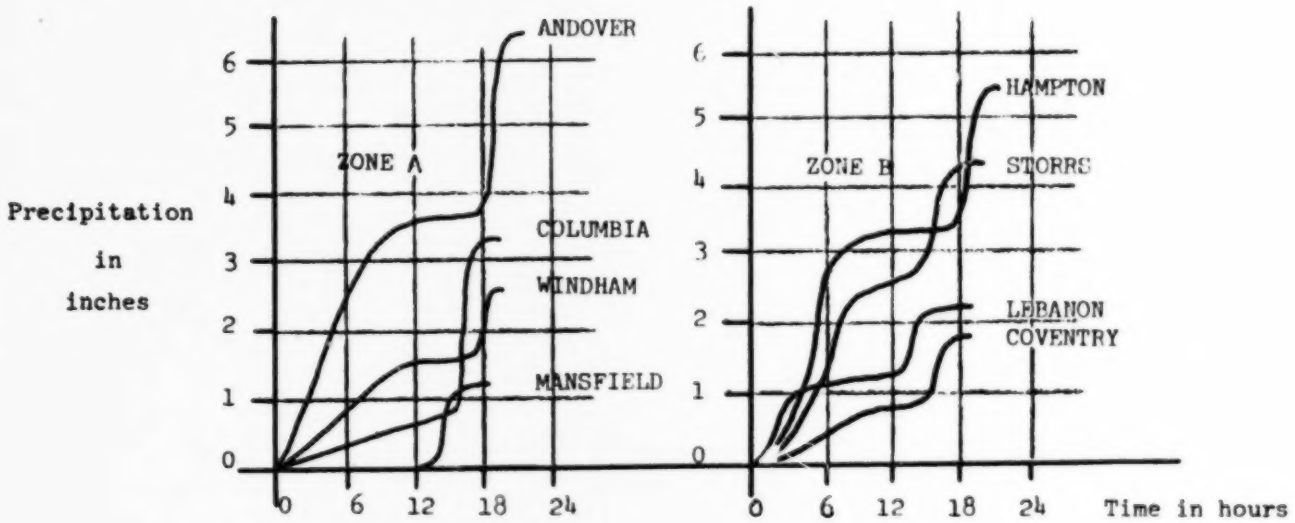


FIG. 12 EXAMPLE, DEPTH-AREA-DURATION CURVES: HYDRO FLOW CHART

A) MASSCURVES OF RAINFALL:



B) ISOHYETAL MAP:

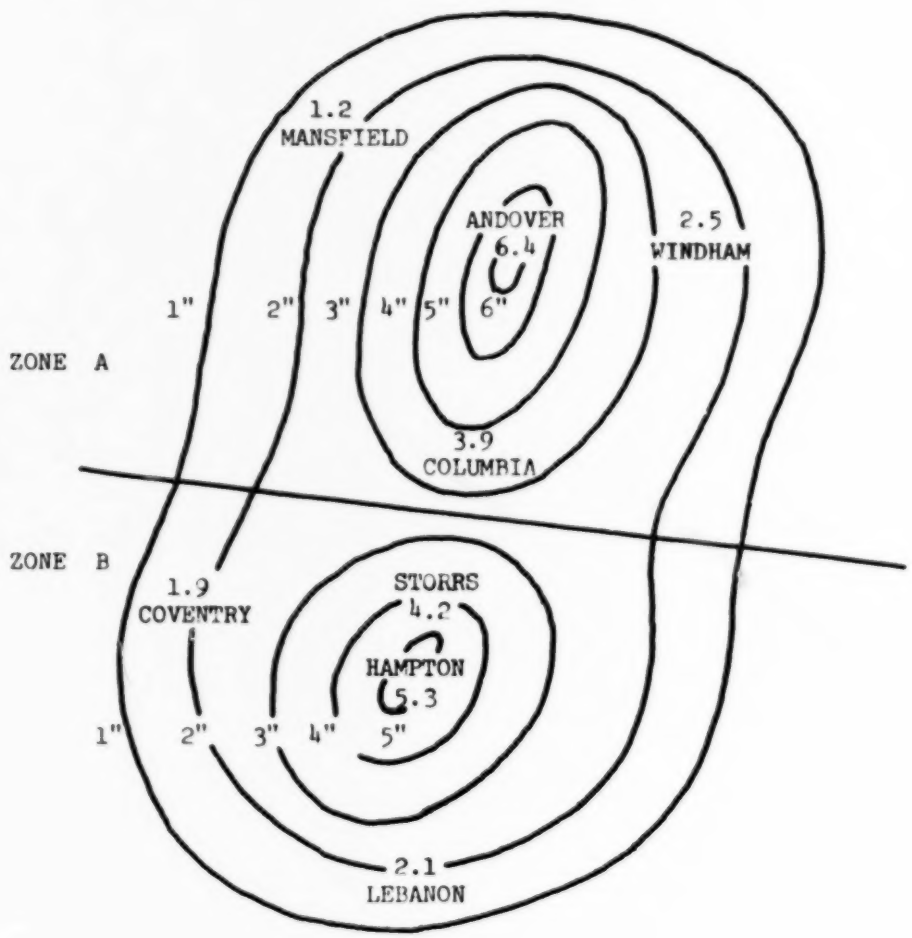


FIG. 13 EXAMPLE, DEPTH-AREA-DURATION CURVES: MASSCURVE AND ISOHYET DATA

```
SIX.HOUR.MASSCURVES
   TOTAL11 ← 8;      TOTAL10 ← 20;
   ZONE1 ← *A*,*A*,*A*,*A*,*B*,*B*,*B*,*B*;
   STATION1 ← *ANDOVER*,*COLUMBIA*,*WINDHAM*,*MANSFIEL*,
      *HAMPTON*,*STORRS*,*LEBANON*,*COVENTRY*;
   MASSINPUT ← 0.5,0.8,1.1,1.5,1.9,2.3,2.7,3.0,3.3,3.4,
               3.4,3.4,3.4,3.4,3.4,3.4,3.4,3.4,3.5,6.4,
               0.2,0.2,0.2,0.3,0.4,0.4,0.5,0.6,0.7,0.8,
               0.8,0.8,0.8,0.8,0.8,0.9,1.5,3.0,3.9,3.9,
               0.2,0.3,0.5,0.7,0.8,1.0,1.2,1.3,1.4,1.5,
               1.6,1.6,1.6,1.6,1.6,1.6,1.7,2.5,2.5,2.5,
               0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
               0.0,0.0,0.0,0.0,0.2,1.1,1.2,1.2,1.2,1.2,
               0.2,0.5,0.8,1.4,2.3,2.8,3.1,3.2,3.2,3.2,
               3.2,3.2,3.2,3.2,3.2,3.2,3.2,3.2,5.2,5.3,
               0.2,0.3,0.4,0.8,1.0,1.4,2.0,2.6,2.8,2.8,
               2.8,2.8,2.8,2.8,2.8,2.8,2.3,3.8,4.2,4.2,
               0.3,0.6,0.7,0.8,1.0,1.1,1.2,1.3,1.3,1.3,
               1.3,1.3,1.3,1.3,1.3,1.8,2.1,2.1,2.1,2.1,
               0.0,0.1,0.2,0.4,0.5,0.7,0.8,0.9,0.9,0.9,
               0.9,0.9,0.9,0.9,1.0,1.9,1.9,1.9,1.9,1.9;
MAXIMUM.STATION.PRECIPITATION
DEPTH.AREA.CURVE
   TOTAL13 ← 15;     CONVERTER ← 1.543;
   CENTER2 ← *A*,*A1*,*A1*,*A1*,*A1*,*A1*,*A1*,*A1*,
             *B*,*B1*,*B1*,*B1*,*B1*,*B1*,*B1*;
   ISOHYET2 ← 0, 6.4, 6, 5, 4, 3, 2, 1,   0, 5.3, 5, 4, 3, 2, 1 ;
   PLANIMETER2 ← 0, 0, 16, 123, 531, 1580, 3652, 6704,
                 0, 0, 36, 319, 1171, 3122, 6749 ;
DEPTH.DURATION.SINGLE.ZONES
   TOTAL14 ← 32;
   ZONE3 ← *A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,*A*,
           *B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B*,*B* ;
   STATION3 ← *ANDOVER*,*ANDOVER*,*ANDOVER*,*ANDOVER*,
              *COLUMBIA*,*ANDOVER*,*COLUMBIA*,*WINDHAM*,
              *ANDOVER*,*COLUMBIA*,*WINDHAM*,*MANSFIEL*,
              *ANDOVER*,*COLUMBIA*,*WINDHAM*,*MANSFIEL*,
              *HAMPTON*,*HAMPTON*,*HAMPTON*,*STORRS*,
              *HAMPTON*,*STORRS*,*LEBANON*,*COVENTRY*,
              *HAMPTON*,*STORRS*,*LEBANON*,*COVENTRY*,
              *HAMPTON*,*STORRS*,*LEBANON*,*COVENTRY* ;
   ISOHYET3 ← 0, 6, 5, 4, 0, 3, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0,
              0, 5, 4, 0, 3, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0 ;
   PLANIMETER3 ← 0, 16, 125, 412, 120, 833, 558, 182, 1034, 1338,
                 1122, 159, 0, 0, 0, 0, 0, 36, 211, 108, 576, 566, 22, 10,
                 798, 1230, 728, 357, 0, 0, 0, 0 ;
DEPTH.DURATION.COMBINED.ZONES
   TOTAL15 ← 6 ;
   ZONE4 ← *A*,*B*,*A + B*,*A*,*B*,*A + B* ;
   ISOHYET4 ← 2, 2, 2, 1, 1, 1 ;
   PLANIMETER4 ← 1, 1, 1, 0, 0, 0 ;
DEPTH.AREA.DURATION.CURVES
PROGRAM.END
```

FIG. 14 EXAMPLE, DEPTH-AREA-DURATION CURVES: HYDRO PROGRAM

MAXIMUM DEPTH - AREA - DURATION CURVES

AREA IN SQ.MI.	MAXIMUM DEPTHS FOR EACH DURATION AND AREA									
	DURATION IN HOURS									
	6	12	18	24	30	36	42	48	54	60
20637	1.1	1.2	1.8	2.3						
10391	1.5	1.6	2.4	3.0						
10353	0.9	1.2	1.6	2.2						
10284	1.3	1.5	2.0	2.4						
5602	1.7	1.9	2.7	3.1						
4789	1.2	1.7	2.1	3.0						
2424	2.1	2.3	3.3	3.9						
1796	1.4	2.4	2.5	3.8						
815	2.2	2.6	3.9	4.8						
489	1.7	2.8	2.8	4.6						
189	2.4	2.9	4.5	5.6						
55	2.2	3.1	3.1	5.1						
14	2.9	3.3	4.7	6.2						
11	2.4	3.5	5.2	6.1						
10	3.0	3.4	5.5	6.4						
10	2.9	3.2	4.4	5.3						
10	2.4	3.6	5.3	6.2						

FIG. 15 EXAMPLE, DEPTH-AREA-DURATION CURVES: OUTPUT

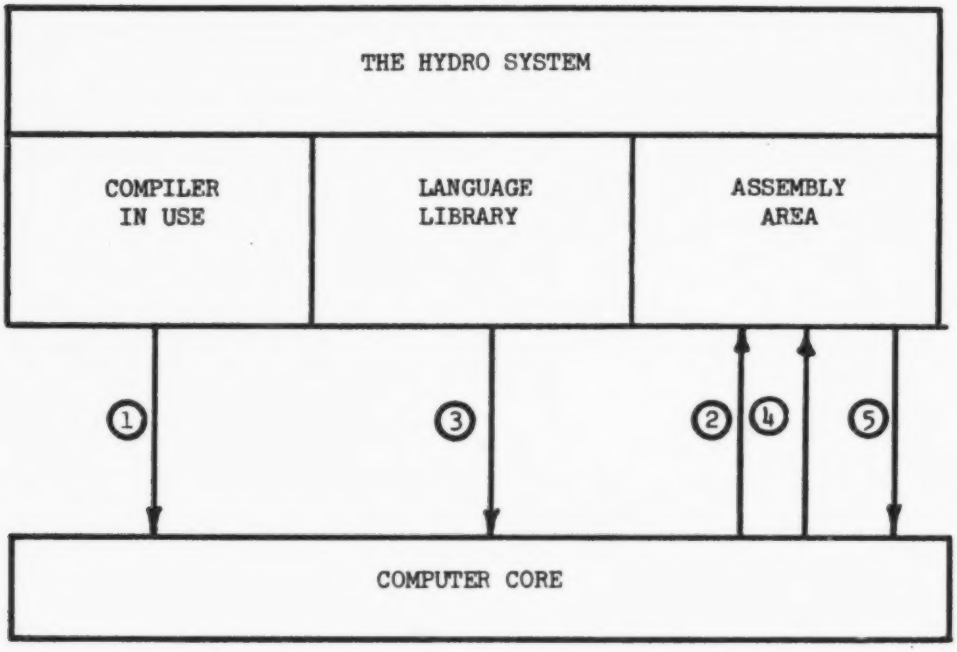
THE HYDRO SYSTEM			
COMPILER IN USE	LANGUAGE LIBRARY	ASSEMBLY AREA	COMPILER IN STORAGE
DISC-TAPE SEVEN SECTIONS CODED IN ALGOL ARRANGED FOR USE WHEN A HYDRO PROGRAM IS SUBMITTED	DISC-TAPE HYDRO SUBROUTINES CODED IN ALGOL MATERIAL IS DRAWN FROM VARIOUS PARTS OF THE LIBRARY WHEN A HYDRO PROGRAM IS RUN	DISC-TAPE GENERAL WORK-SPACE USED FOR ASSEMBLY OF ALGOL PROGRAM FROM HYDRO CODE	DISC-TAPE SEVEN SECTIONS CODED IN ALGOL USED ONLY WHEN MAKING ALTERATIONS TO THE HYDRO SYSTEM NOT WHEN RUNNING A HYDRO PROGRAM

FIG. 16 THE HYDRO SYSTEM

THE HYDRO SYSTEM			
COMPILER IN USE	LANGUAGE LIBRARY	ASSEMBLY AREA	COMPILER IN STORAGE
Section 1: 650 card images 13000 words to run	4000 card images	4000 card images	3500 card images
Section 2: 600 card images 13000 words to run			
Section 3: 550 card images 9400 words to run			
Section 4: 650 card images 12000 words to run			
Section 5: 450 card images 8000 words to run			
Section 6: 500 card images 9200 words to run			
Section 7: 100 card images 2000 words to run			

NOTE: LARGEST ASSEMBLED PROGRAM RUN TO DATE
1000 card images
12000 words of internal storage to run

FIG. 17 DIMENSIONS OF THE HYDRO SYSTEM



1. Compiler loaded into core to execute HYDRO program.
2. Data of HYDRO program stored onto assembly area.
3. Proper sections of ALGOL code drawn from library.
4. ALGOL program composed on the assembly area.
5. Final assembled program loaded into core for execution.

FIG. 18 OPERATION OF THE HYDRO SYSTEM